第 35 卷 第 8 期 计 算 机 工 程 2009 年 4 月
Vol.35 No.8 Computer Engineering April 2009
·人工智能及识别技术· 文章编号：1000—3428(2009)08—0208—03 文献标识码：A 中图分类号：TP312

一种新的蛋白质序列模式挖掘算法

郭 顺，姜青山，王备战，史 亮
(厦门大学软件学院，厦门 361005)

摘 要：针对传统模式挖掘方法挖掘蛋白质序列会生成大量候选模式或多次构造投影数据库，导致效率降低，挖掘过程中会产生不必要的短模式或错误模式等问题，提出基于模式划分的 MBioPM 算法。理论分析和实验表明，MBioPM 算法的性能高于其他相关算法。
关键词：蛋白质序列；模式挖掘；数据挖掘；生物信息学

Mining Algorithm for Protein Sequence Pattern

GUO Shun, JIANG Qing-shan, WANG Bei-zhan, SHI Liang
(Software School, Xiamen University, Xiamen 361005)

【Abstract】Traditional algorithms face efficiency problem because of generating a huge number of candidates or constructing projected database many times. These algorithms will generate unnecessary short patterns or even wrong patterns in the process of mining. To attack these problems, a novel mining algorithm called Motif-divide based Biology sequence Pattern Mining(MBioPM) is presented based on “motif-divide” method. The MBioPM algorithm improves the efficience and avoids the problems mentioned above. Theoretical analysis and experimental results show that MBioPM algorithm improves performance as compared with other algorithm.
【Key words】protein sequence; pattern mining; data mining; bioinformatics

1 概述

生物信息学(bioinformatics)是生命科学、计算机科学、信息科学和数学等学科交汇融合所形成的一门交叉学科[1]，蛋白质序列数据是生物信息学的主要研究对象之一。如何挖掘蛋白质序列中的序列模式对蛋白质分析十分重要。

随着近年来的研究与发展，很多模式发现算法被提出。文献[2]和文献[3]分别针对该研究领域提出具体的观点。由于序列数据的快速增长，这些方法大多面临效率问题。文献[4]提出多支持度的概念以及基于前缀投影数据库的 BioPM 算法，该算法克服了传统算法的缺点，在性能和效率方面有明显改善。然而，这些算法都会在挖掘过程中产生不必要的短模式或错误的模式，产生大量的候选集或多次构造投影数据库，因此，算法效率有待提高。

本文提出一种新算法 MBioPM(Motif-divide based Biology sequence Pattern Mining)，引入模式划分的方法，从指定较长的模式长度开始挖掘，避免产生不必要的短模式，分析和实验表明，MBioPM 算法的性能高于其他相关算法。

2 相关工作

传统的生物序列模式挖掘过程[4]如图 1 所示。

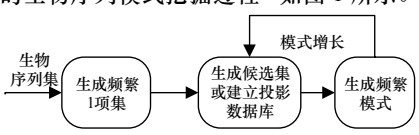

图 1 生物序列模式挖掘过程

模式挖掘的具体步骤如下：

(1)扫描序列数据库，找到所有序列的频繁 1 项集。

(2)根据生成的频繁项集，产生候选频繁集或建立投影数据库，前者主要是类 Apriori 算法，后者包括 BioPM 算法。

(3)将候选模式或投影数据库中的模式进行匹配生成频繁模式。

(4)进行模式增长，重复步骤(2)、步骤(3)。

从上述步骤可以看出，由于采用了自底向上宽度优先策略挖掘所有频繁模式，因此传统算法都需要进行多次的模式增长才能得到较长的频繁模式，而太短的模式对生物序列没有多大意义。

为满足生物序列挖掘的特定需求以及进一步提高算法效率和性能，文献[4]引进多支持度概念，并提出基于前缀投影数据库的 BioPM 算法。其多支持度概念的相关定义如下：

定义 1 蛋白质序列[4] 蛋白质序列可看作数量为 r 的有限字符集Σ上的字符串。

$\Sigma=\{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,S,T,V,W,Y\}$

一条蛋白质序列记作 S，由 M 条蛋白质序列构成的序列集记为 D，$D=\{S_1, S_2,\cdots, S_M\}$。

定义 2 局部支持度[4] 给定序列集 D，在 D 的某一序列 S 中子序列 T 出现的次数称为局部支持度。子序列 T 的局部支持度是 S 中包含子序列 T 的数目，记为 $loc\_sup_D(T)$。

定义 3 分布支持度[4] 给定序列集 D，在 D 中包含子序列的序列数称为分布支持度。子序列 T 的分布支持度是 D 中包含子序列 T 的序列数，记为 $dis\_sup_D(T)$。

由于引入了多支持度概念，因此 BioPM 算法能满足多种挖掘需求，包括保守模式挖掘、重复模式挖掘及两者结合的

基金项目：国家自然科学基金资助项目(10771176)；厦门大学校科研基金资助项目(0630-X0117)
作者简介：郭 顺(1982－)，男，硕士研究生，主研方向：数据挖掘；姜青山，教授、博士生导师；王备战，教授；史 亮，副教授
收稿日期：2008-10-20 **E-mail：**gsgowell@gmail.com

挖掘。再者，BioPM 算法基于前缀投影数据库，不会产生候选模式，并且设置了一个窗口值 w 来控制模式增长，使算法的性能和效率进一步提高。

BioPM 算法的过程如下：

(1)扫描序列数据库 D，找到 D 中的频繁 1 项集合 $fre1$。

(2)对于 $fre1$ 中的每个频繁 1 项集 α，做如下操作：

1)构建 α 的投影数据库 $D|\alpha$；

2)建立序列模式树，根节点的值为 α；

3)调用 $subBioPM(\alpha,l,D|\alpha)$。

$subBioPM(\alpha,l,D|\alpha)$是一个递归调用的过程，其步骤如下：

(1)扫描 $D|\alpha$，调用 *insertBioPTree* 找到跨度为 w 的频繁片段 b。

(2)对于每个频繁模式 b，加上前缀 α，形成序列模式 α'，输出。

(3)对于每个 α'，构建其投影数据库，调用 $subBioPM(\alpha', l+w,D|\alpha')$。

$insertBioPTree(\alpha, l, D|\alpha)$的步骤如下：

(1)对每个 $D|\alpha$ 中的序列的前缀长为 w 的子序列 *tempb* 做如下操作：

1)比较 BioP-Tree 路径为 α 的所有叶子节点；

2)找到匹配的节点，如果是同一序列就将局部支持度加 1，否则将分布支持度加 1；

3)没找到匹配节点，将 *tempb* 作为新的叶子节点。

(2)扫描所有叶子节点 v，如果其支持度大于阈值，则将其加入频繁模式。

从算法的流程中可以看出，BioPM 算法依然是从频繁 1 项集开始构建投影数据库，该步骤需要构建大量投影数据库，开销很大，尤其当支持度阈值较低时。事实上，太短的模式对蛋白质序列没有意义，而 BioPM 算法可能需要模式增长几次才能挖掘到长度较长的频繁模式。另外，BioPM 算法引入了间隔阈值，但依然可能挖掘到错误的模式。

根据以上分析，本文提出了 MBioPM 算法，该算法能够直接从一个较长的指定模式长度开始挖掘，提高了算法效率，并且避免了产生大量不必要的短模式。

3 MBioPM 算法

MBioPM 算法采用上述 BioPM 算法的多支持度概念定义，使 MBioPM 算法同样能进行多支持度挖掘。

3.1 MBioPM 算法原理

MBioPM 算法的主要思想是从一个给定的较长的长度开始挖掘该长度下数据集中的所有频繁模式，然后进行模式长度增长。为了挖掘某个长度下的频繁模式，MBioPM 算法引入了划分的方法。

划分的过程如下：对每一条长度为 n 的生物序列 seq，根据模式长度 k 进行 k 次划分，每次划分从 seq 的第 $i\ (0 \leqslant i \leqslant k-1)$ 个位置开始，将 seq 分成 $\lfloor (n-i)/k \rfloor$ 个连续的模式。显然，划分结束后得到的模式总数不大于 n。

划分结束后，得到该生物序列长度为 k 的所有模式，对这些模式进行匹配，就能得到所需要的频繁模式。

3.2 MBioPM 算法过程

MBioPM 算法主要是在 3.1 节所描述的划分过程中挖掘频繁模式，具体过程如下：

例如，生物序列片段 seq 为

<AGTTCTAACAGGAAGACGT>

按照模式长度 4 进行分段，共进行 4 次划分。第 i 次划分从 seq 的第 i 个位置开始，每 4 个字符作为一个长度为 4 的模式。对于划分到 seq 末尾长度不到 4 的片段，舍去。划分完后得到如图 2 所示的模式。

Motif			
AGTT	CTAA	CAGG	AAGA
GTTC	TAAC	AGGA	AGAC
TTCT	AACA	GGAA	GACG
TCTA	ACAG	GAAG	ACGT

图 2　序列 *seq* 按照模式长度 4 划分得到的模式

为了挖掘得到长度为 4 的频繁模式，在划分过程中进行如下操作：

(1)建立一个缓存区，每次划分得到一个模式，先查找缓存区中是否有模式与之匹配。

(2)如果匹配，看缓存区中的模式是否与该模式属于同一条序列，如果是，将缓存区中的模式的局部支持度 *loc_sup* 加 1，并判断该值得是否大于最小支持度阈值，若大于阈值，则将该模式加入局部频繁集中。

(3)如果不属于同一条序列，将缓存区的分布支持度 *dis_sup* 加 1，同时判断该值得是否大于最小支持度阈值，大于就将该模式加入分布频繁集合中。

(4)如果模式与缓存区模式不匹配，则将该模式加入缓存区中。

对序列集合 D 中的每条序列进行以上操作后，第 1 次挖掘就得到了 D 中所有长度为 4 的频繁模式。

而后，清空缓存区，进行模式长度增长，即按照新的模式长度进行划分，再重复以上步骤。不同的是，模式匹配时，先将模式前缀和先前得到的频繁模式进行比较。如果模式前缀和频繁模式匹配，再和缓存区中片段进行匹配。否则，直接跳到下一模式。

从算法过程可以看到，MBioPM 算法直接从一个指定的较长的模式长度开始挖掘，这使算法更加具有针对性，同时避免了生成大量的短模式，提高了算法效率。另外，MBioPM 算法在模式增长过程中既不会生成大量的候选模式，也不需要大量构建投影数据库。

3.3 MBioPM 算法的主要参数

模式挖掘起始长度 a：从模式长度 a 开始挖掘，然后进行模式增长。为进一步提高算法的效率，也可以设置一个窗口值 w 来控制模式增长。

最小支持度阈值 r：模式支持度大于等于该值的模式为频繁模式

MBioPM 算法伪代码如下：

输入 生物序列集合 D，模式挖掘起始长度 a，最小支持度阈值 r，窗口值 w

输出 频繁模式集合

```
for i := a;; i+ = w; do //模式增长
扫描数据库 D，对 D 中每条序列 S do Divide(S, i)
//将 S 按模式长度 i 进行划分
每划分得到一个模式 motif
Compare (prefixmotif, fre(i-1))
//和频繁集比较，若是第一次就直接跳到查找缓存区 Buffer
if 找不到匹配
跳到 S 划分的下一个模式
else {
查找缓存区 Buffer
if motif Match Buffer.bmotif
```

```
//找到匹配的模式
if motif.id == Buffer.bmotif.id
//属于同一序列
        Buffer.bmotif.loc_sup ++;
        if Buffer.bmotif.loc_sup>=r
           fre_loc.add(Buffer.bmotif);
//加入频繁集
else Buffer.bmotif.dis_sup ++;
 //不属同一序列
        if Buffer.bmotif.dis_sup>=r
           fre_dis.add(Buffer.bmotif);
//加入频繁集
    Buffer.bmotif.id = motif.id;
跳到 S 划分的下一个模式;
    else 找不到 Buffer 中模式与 motif Match,
        Buffer.add (motif);
//Buffer 加入新的模式
跳到 S 划分的下一个模式; }
扫描 D 完毕,清空缓存区 Buffer
如果没有生成新的频繁模式,return
end for
```

4 实验与结果分析

为了测试 MBioPM 算法的性能和效率，选取了 Apriori 算法和 BioPM[4]算法在 Pfam[5]数据集上进行比较。

4.1 实验数据

实验数据采用真实的来自 Pfam(http://pfam.sanger.ac.uk/)蛋白质家族数据库中的 10 个蛋白质家族的序列(序列平均长度为 800)[5]。

4.2 实验环境

本实验环境为 3.00 GHz Pentium 4 CPU，1.00 GB 内存，WinXP 系统，JCreator Pro3.50。算法用 Java 语言实现。实验参数设置：起始挖掘长度 *a* 为 6，窗口值 *w* 为 2。

4.3 运行性能

实验的目的是对比 3 种算法在不同阈值支持度下的运行时间。实验样本取自 Pfam 中 3 个家族(Globin, Glyco_hydro_19, 7kD_viral coat)，从各家族序列中取一部分作为测试集(共 50 条)。参与比较的算法为 Apriori[5]算法和 BioPM 算法。

实验结果表明，MBioPM 运行时间明显较少，尤其在阈值较小的情况下。这是因为在支持度阈值较小的情况下，满足条件的短模式大大增加。Apriori 算法生成了更多的短模式及候选模式，大大增加了算法开销，见图 3。

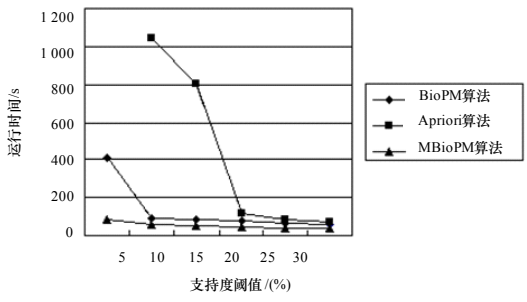

图 3　3 种算法运行时间对比

BioPM[4]算法同样因为大量的短模式及多次构建投影数据库影响了算法效率。相反，MBioPM 算法由于避免了生成大量的短模式以及候选模式，提高了效率。

4.4 时间效率比较

本实验的目的是检查在固定的支持度阈值下，随着序列数的增长，MBioPM 算法和 BioPM 算法运行时间比较。实验序列样本来自 Pfam 数据库中 10 个家族(Globin, short chain dehydrogenase, SBP_bac_9, Acety- -ltransferase, GNAT family, ATPase family, 2OG-Fe(II) oxygenase, 2'5'RNA ligase family, Glyco_hydro_19, Mycop- -lasma haemagglutinin, 7kD_viral coat 等)，从各个家族序列中分别抽取其中一部分，支持度阈值取 15%。实验从 100 条序列开始逐渐增加，2 种算法的运行时间比较见图 4。

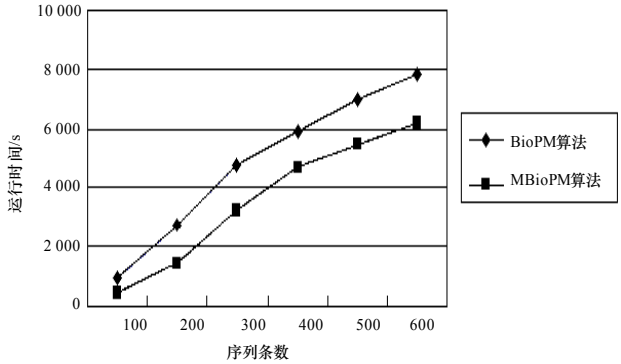

图 4　MBioPM 与 BioPM 算法运行时间对比

实验结果表明，在固定支持度阈值的情况下，2 种算法的运行时间随着序列数目的增加而增加，但是，MBioPM 算法的运行时间比 BioPM 算法少，因为 MBioPM 算法节省了生成短模式的时间开销。

5 结束语

本文在多支持度概念的基础上提出了一种新的基于划分的蛋白质模式挖掘算法 MBioPM。该算法能从长度较长的模式开始挖掘，使挖掘更加具有针对性，同时避免了产生大量不必要的短模式，提高了算法效率。分析和实验表明，MBioPM 算法能有效挖掘蛋白质序列中的保守模式，尤其对于支持度阈值较少的情况，有较好的效果。

参考文献

[1] Luscombe N M, Greenbaum D, Gerste M. What Is Bioinformatics: A Proposed Definition and Overview of the Field[J]. Methods Information in Medicine, 2001, 40(4): 346-358.

[2] Brazma A, Jonassen I, Eidhammer I, et al. Approaches to the Automatic Discovery of Patterns in Biosequences[J]. Journal of Computational Biology, 1998, 5(2): 279-305.

[3] Brejova B, DiMarco C, Vinar T. Finding Patterns in Biological Sequences[M]. [S. l.]: IEEE Press, 2000.

[4] Xiong Yun, Zhu Yangyong. BioPM: An Efficient Algorithm for Protein Motif Mining[C]//Proc. of ICBBE'07. [S. l.]: IEEE Press, 2007.

[5] Bateman A, Birney E, Cerruti L, et al. The Pfam Protein Families Database[J]. Nucleic Acids Res., 2002, 30(1): 276-288.

编辑　张正兴